Projet groupe 10 :

Voiture télécommandée LEGO avec un Raspberry Pi Pico W

Participants :

Sean Vergauwen

Edward Gay

Noah Cavrenne

Baris Ozcelik

I. Introduction

Contexte du Projet :

Dans le cadre du cours d'électronique digitale de cette année, nous avons été invités à réaliser un projet intégrant les concepts vus en théorie : logique combinatoire, microcontrôleurs, communication, et interfaces. Le but de ce projet était de mettre en œuvre ces compétences de manière concrète à travers la conception d'un système embarqué. Notre groupe a choisi de développer une voiture LEGO télécommandée à l'aide d'un Raspberry Pi Pico W, en combinant électronique, programmation et mécanique.

Notre groupe a été particulièrement motivé par l'idée de construire une voiture télécommandée en LEGO, car cela nous permettait de combiner plusieurs domaines que nous apprécions : la programmation embarquée, la conception électronique, et l'assemblage mécanique. Le choix d'utiliser des pièces LEGO, à la fois modulables et accessibles, offrait une liberté créative dans la conception du châssis. De plus, l'aspect interactif du projet – pouvoir contrôler un objet réel à distance a renforcé notre intérêt pour sa réalisation. Ce projet représentait un excellent moyen de mettre en pratique nos acquis tout en relevant un défi technique stimulant.

Présentation succincte des principales technologies utilisées : Raspberry Pi Pico W, MicroPython, L298N, HC-SR04, interface web, Express, MongoDB, Nginx, Docker

Raspberry Pi Pico W

- Microcontrôleur basé sur le RP 2040.
- Intègre une connectivité Wi-Fi
- Utilisé pour contrôler les moteurs, lire les capteurs et gérer la communication avec le serveur via TCP.

MicroPython

- Langage de programmation dérivé de Python, adapté aux microcontrôleurs.
- Léger, simple et rapide à mettre en œuvre sur le Pico W.
- Permet de gérer les entrées/sorties, le capteur, les moteurs, et la connexion réseau.

L298N (Contrôleur de moteurs)

- Module double pont en H permettant de contrôler deux moteurs à courant continu.
- Gère le sens de rotation via des signaux envoyés par le Pico W.
- Indispensable pour le déplacement avant, arrière, gauche, droite.

HC-SR04 (Capteur à ultrasons)

- Mesure la distance d'un obstacle en utilisant des ondes ultrasoniques.
- Intégré pour activer automatiquement le freinage lorsqu'un obstacle est détecté à une distance critique. (12cm)
- Connecté au Pico W via des broches GPIO.

Interface Web

- Application HTML/CSS/JavaScript permettant de piloter la voiture à distance.
- Contient des boutons directionnels (haut, bas, gauche, droite, stop).
- Envoie des commandes à l'API via des requêtes HTTP POST.
- Affiche en temps réel l'état de la connexion avec le Pico W.
- Disponible à cette adresse : https://voiture.seanvergauwen.com

Express (API et serveur TCP)

- Interface de communication entre le frontend (page web) et le Raspberry Pi Pico W.
- Gère les commandes reçues du client web et les transmet au Pico via TCP.
- Contient des endpoints pour les commandes (`/api/command`), pour le statut de la voiture (`/api/status`) et (`/api/commandCounts`) pour récupérer les compteurs d'actions.

MongoDB (Base de données NoSQL)

- Utilisée pour stocker l'historique des interactions avec la voiture.
- Enregistre les commandes envoyées.
- Permet un suivi de l'activité et une analyse a posteriori.

Nginx

- Permet de servir l'interface web sur internet.
- Est aussi utilisé pour gérer les certificats TLS pour l'HTTPS.
- Utilisé comme reverse proxy pour rediriger les calls d'API vers celle-ci.

Docker

- Permet de conteneuriser nos différents services.
- Permet également de configurer un système d'exploitation et des versions de logiciels différentes sans avoir à gérer les problèmes de dépendance.
- Règle le fameux problème du "ça fonctionne sur mon pc pourtant !"

- ## Objectifs du Projet :

Concevoir une voiture LEGO mobile capable de se déplacer dans toutes les directions (avant, arrière, gauche, droite) à l'aide de moteurs DC pilotés par un Raspberry Pi Pico W. **Permettre un contrôle à distance** de la voiture via une interface web intuitive, accessible depuis un ordinateur connecté au même réseau. **Implémenter un système de freinage automatique** en intégrant un capteur à ultrasons (HC-SR04), capable de détecter un obstacle et d'arrêter le véhicule à une distance prédéfinie. **Assurer une communication efficace** entre l'interface utilisateur et la voiture à l'aide d'un protocole TCP sécurisé. **Enregistrer l'activité de la voiture** (commandes) dans une base de données MongoDB pour permettre un suivi et une analyse des interactions. **Concevoir un PCB fonctionnel** pour centraliser les connexions des composants électroniques et faciliter leur intégration dans le châssis.

- ## Structure du Rapport :

Ce rapport est structuré en cinq grandes parties : une introduction présentant le contexte et les objectifs du projet ; une description détaillée de la conception matérielle et logicielle du système ; une section dédiée à la mise en œuvre, aux tests et aux validations ; un chapitre sur la gestion de projet et la collaboration au sein du groupe ; et enfin, une conclusion avec un bilan global et des perspectives d'amélioration.

II. Conception et Architecture du Système

Dans cette section, nous détaillons l'architecture complète de notre système, en abordant à la fois les aspects matériels et logiciels. Nous présentons les choix techniques réalisés pour la structure du véhicule, le schéma électronique, la conception du PCB, ainsi que le développement du code embarqué, de l'interface web, de l'API et de la base de données.

III. Mise en Œuvre et Tests

Cette partie retrace les étapes de développement et d'intégration du projet. Elle décrit notre méthodologie de travail, les tests réalisés sur les différents modules (moteurs, capteur, communication, interface), ainsi que les résultats obtenus. Nous évaluons ici dans quelle mesure le système répond aux exigences fixées initialement.

IV. Gestion de Projet et Collaboration

Cette section revient sur l'organisation interne du groupe tout au long du projet. Nous y expliquons la répartition des tâches, les outils de communication utilisés, la gestion des échéances, ainsi qu'un retour critique sur la collaboration entre les membres de l'équipe.

V. Conclusion et Perspectives

Enfin, nous faisons un bilan global du projet, en soulignant les points forts, les difficultés rencontrées et les solutions apportées. Nous identifions également les limites du système actuel et proposons plusieurs pistes d'amélioration pour de futures itérations.

II. Conception et Architecture du Système

- ## Conception Matérielle :

Châssis et Assemblage Lego :

La structure mécanique de notre voiture repose entièrement sur des **éléments LEGO Technic**, choisis pour leur modularité, leur solidité et leur facilité d'intégration des composants électroniques.

Structure de base :

La base du châssis est constituée de **pièces LEGO Technic**, renforcées par des plaques transversales pour assurer la rigidité de l'ensemble. Nous avons utilisé des **pièces issues de beaucoup de sets LEGO différents**, notamment pour les supports moteurs et les éléments de direction. La voiture repose sur **quatre roues**, avec les deux roues arrière motorisées et les deux roues avant passives pour faciliter la direction.

Intégration des moteurs :

Les moteurs LEGO Technic DC sont montés à l'arrière du châssis et reliés directement aux roues via des engrenages. Cette configuration permet une **propulsion différenciée**, essentielle pour la direction (tourner en faisant varier la vitesse des deux roues arrière).

Positionnement des composants électroniques :

Le **Raspberry Pi Pico W** est monté à l'arrière du châssis sur le **PCB**, fixé avec des éléments Technic adaptés. Le **module L298N** est placé au milieu, à proximité des moteurs, afin de réduire la longueur des fils. Le **capteur à ultrasons HC-SR04** est installé à l'avant de la voiture, bien centré, pour détecter les obstacles sur le trajet.

Gestion des câbles :

Les fils sont passés proprement dans les interstices du châssis LEGO, et regroupés à l'aide de **colsons** ou d'éléments LEGO flexibles. Cette gestion permet d'éviter tout enchevêtrement ou frottement contre les roues, tout en conservant un aspect modulaire.

Avantages du choix LEGO :

Flexibilité, les pièces peuvent être facilement remplacées ou positionnées en fonction des ajustements nécessaires. **Accessibilité**, permet aux membres de l'équipe de modifier la structure sans outil ni équipement spécialisé. **Compatibilité avec les moteurs LEGO** déjà disponibles, ce qui réduit les coûts et les besoins d'adaptateurs mécaniques.

Carte Électronique (PCB) :

Le schéma électronique de notre PCB a été réalisé avec les composants suivants :

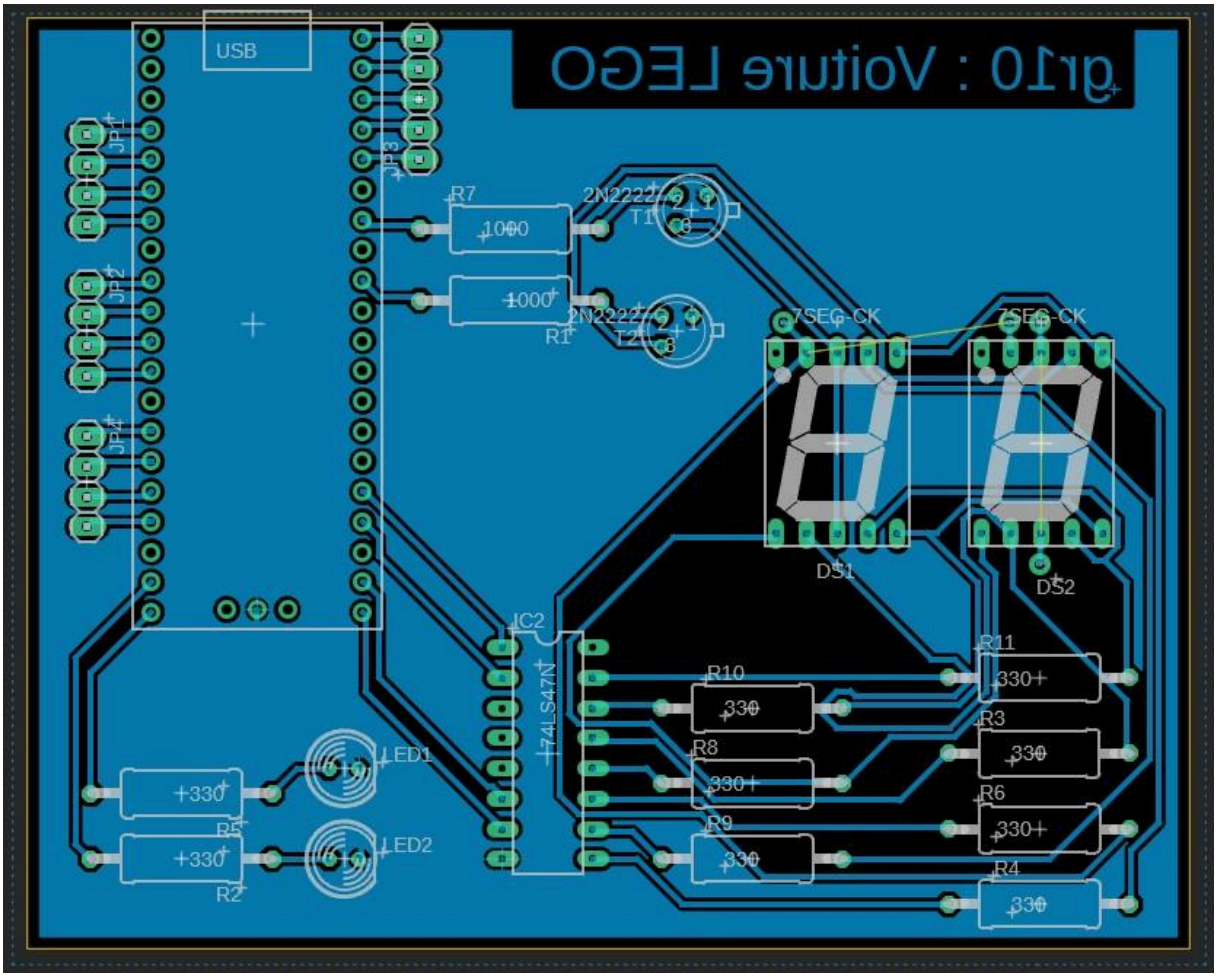
- Un Raspberry Pi Pico servant de microcontrôleur
- Un décodeur 74LS47N
- Deux transistors NPN
- Deux afficheurs 7 segments
- Deux LED
- Une série de résistances
- Plusieurs connecteurs femelles

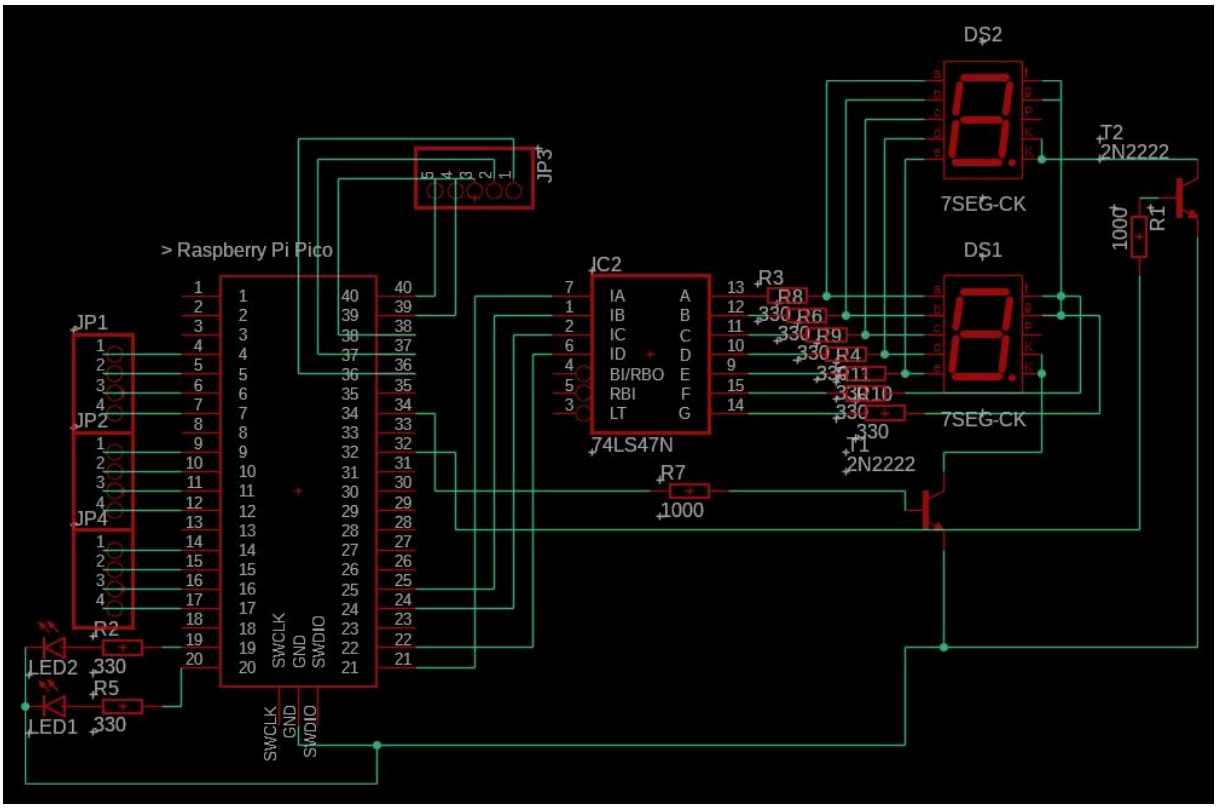
Pour faire fonctionner les deux afficheurs 7 segments avec un seul décodeur 74LS47N, nous avons utilisé la technique du **multiplexage**. Les broches d'entrée du décodeur (IA, IB, IC, ID) sont reliées à des broches GPIO du microcontrôleur. Les sorties du décodeur sont connectées en parallèle aux segments des deux afficheurs, chaque ligne étant protégée par une résistance de 330 ohms.

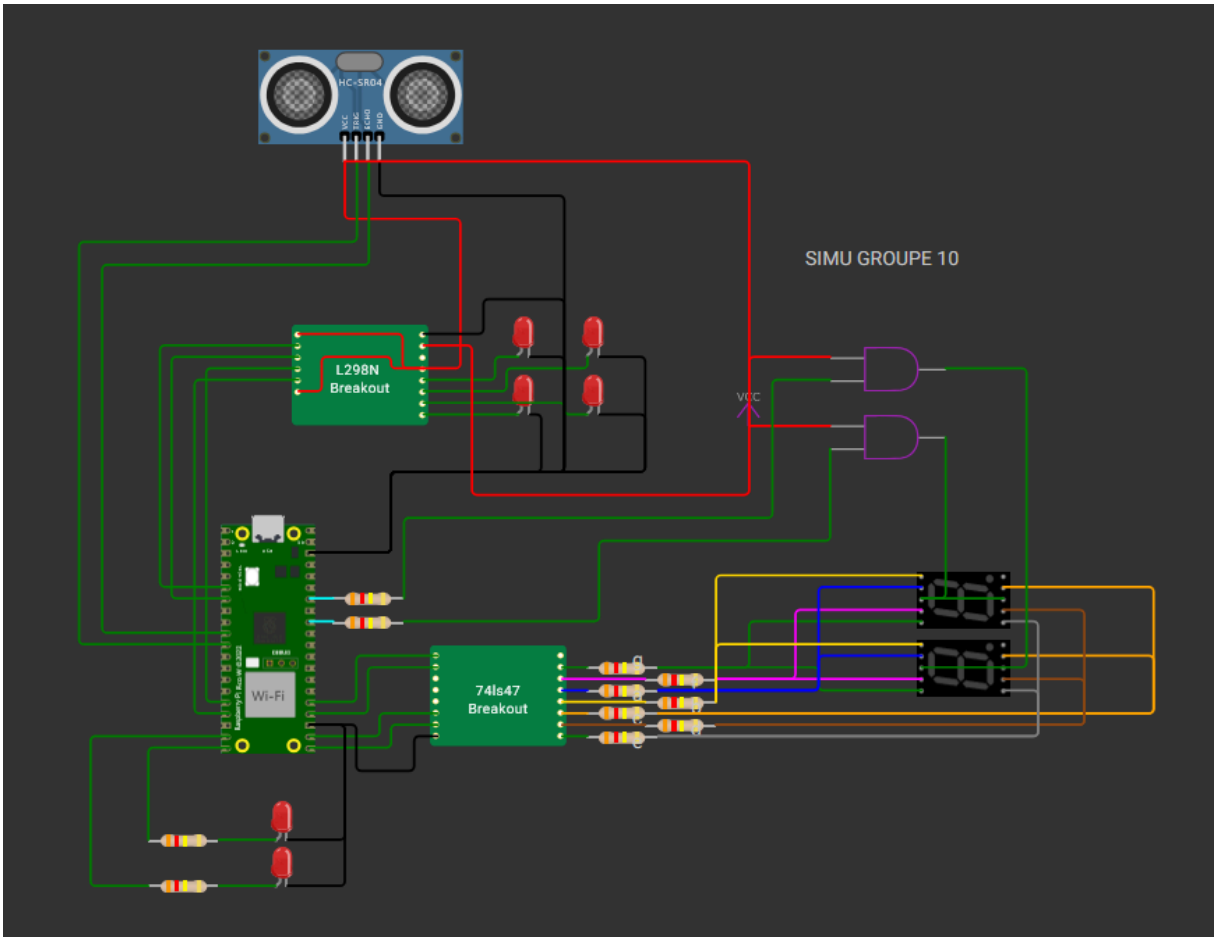
Les **anodes communes** des afficheurs sont chacune reliées à l'alimentation 5V via un transistor NPN, permettant d'activer un afficheur à la fois. Les bases de ces transistors sont commandées par des broches GPIO à travers des résistances de limitation.

Deux LED ont également été intégrées au schéma. Elles sont reliées à des GPIO du microcontrôleur et protégées par des résistances de 330 ohms.

Enfin, des **connecteurs femelles** ont été ajoutés pour permettre le branchement de composants externes.


USB
JP1
JP2
JP4
JP3
R7
1000
2N2222
T1
C8
R1
T2
7SEG-CK
7SEG-CK
DS1
DS2
IC2
74LS47N
R10
330
R11
330
R3
330
R8
330
R6
330
R9
330
R4
330
LED1
LED2
R5
330
R2
330


DS2
7SEG-CK
T2
2N2222
1000
R1
DS1
7SEG-CK
JP3
> Raspberry Pi Pico
JP1
JP2
JP4
IC2
IA
IB
IC
ID
BI/RBO
RBI
LT
A
B
C
D
E
F
G
74LS47N
R3
R8
R6
R9
R4
R11
R10
330
T1
2N2222
R7
1000
R2
330
LED2
R5
330
LED1
SWCLK
GND
SWDIO



Le circuit imprimé (PCB) a été entièrement conçu à l'aide du logiciel Fusion 360. Nous avons d'abord réalisé le schéma électronique en intégrant les différents composants et en les connectant correctement. Une fois cette étape terminée, nous avons lancé le routage automatique du schéma, puis corrigé les éventuelles erreurs en repositionnant certains composants. Comme quelques composants se trouvaient encore sur le trajet des liaisons, nous avons ajouté des vias afin de pouvoir tirer des fils directement sur le PCB.

Nous avons rencontré certains défis lors de la soudure et de l'assemblage des différents composants. Dans un premier temps, des connecteurs femelles trop étroits ont été soudés pour connecter le Raspberry Pi Pico. Nous avons donc dû les dessouder, puis souder des connecteurs plus larges. Les LED ont également dû être ressoudées afin qu'elles soient bien fixées et ne bougent plus. Le reste des composants a été soudé sans problème.

Interface avec les Moteurs (L298N) :

Le **L298N** est un **double pont en H** permettant de contrôler deux moteurs DC de manière indépendante. Il permet de faire tourner un moteur dans les deux sens (avant/arrière) et de faire pivoter les roues avant (gauche/droite) avec l'autre moteur

Fonctionnement :
Chaque moteur est connecté à deux entrées IN1/IN2 et IN3/IN4. En mettant l'une des entrées à HIGH et l'autre à LOW, le moteur tourne dans un sens. En inversant les signaux, le moteur tourne dans l'autre sens.

Système de Détection d'Obstacles (HC-SR04) :

Le **HC-SR04** mesure la distance d'un objet grâce à des ultrasons.

Principe de fonctionnement :

- Le Pico envoie un signal de 10µs sur la broche **TRIG**.
- Le capteur émet une onde ultrasonore.
- L'onde rebondit sur un obstacle et revient.
- Le capteur met la broche **ECHO** à HIGH pendant la durée aller-retour.
- Le Pico mesure cette durée `duration = time_pulse_us(self.echo, 1, Config.ECHO_TIMEOUT_US)`
- Le pico la convertit ensuite en cm `distance_cm = pulse_time * 0.0171`

Utilisation pour le freinage automatique :

Si la distance mesurée < seuil (ex : 12 cm), le Pico coupe l'alimentation des moteurs pour arrêter la voiture.

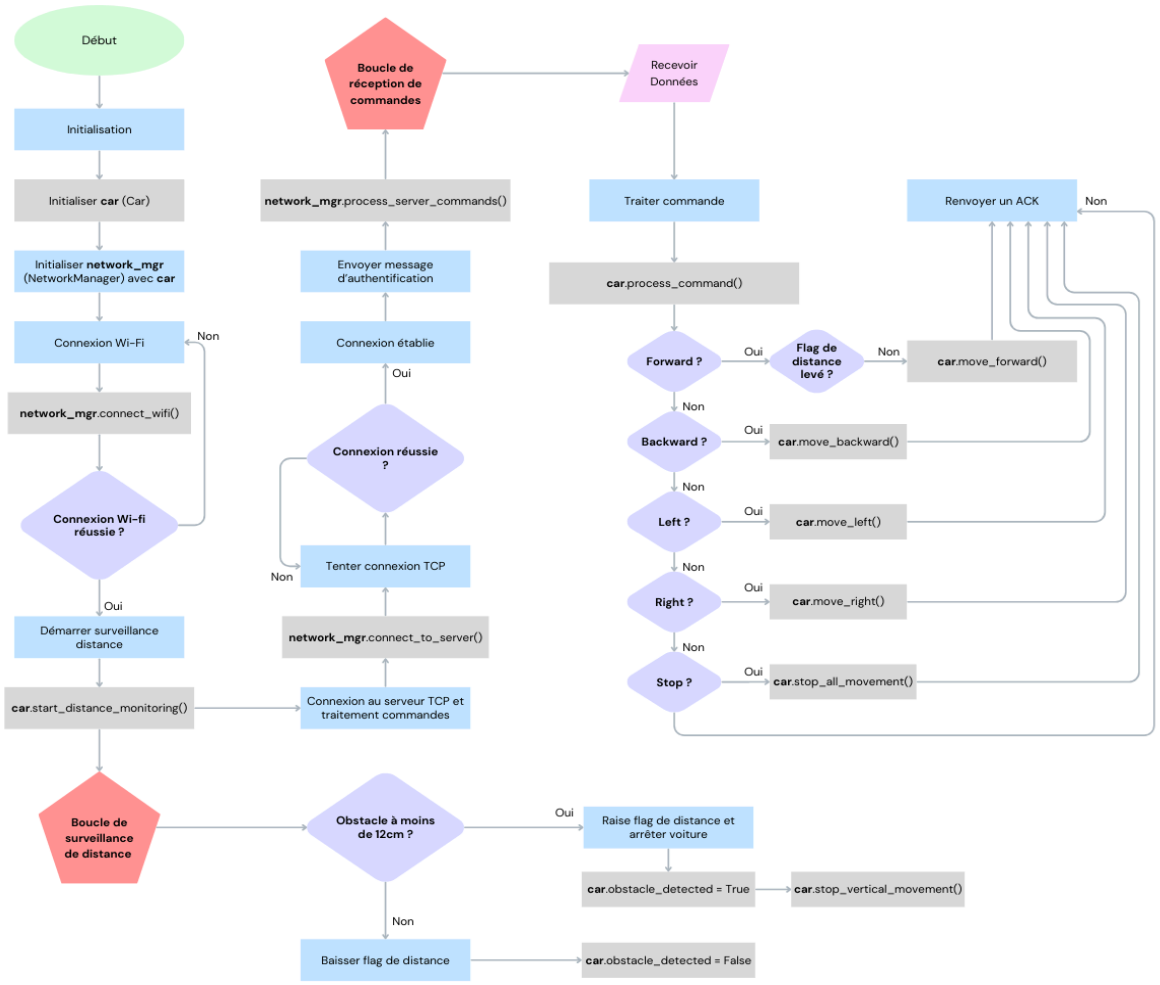
- ## Conception Logicielle :

Code MicroPython (Raspberry Pi Pico W) :

Le **Pico W** initie une **connexion TCP** vers un serveur (backend de l'API).

Étapes :

- Le Pico se connecte au Wi-Fi avec un mot de passe WPA2-PSK.
- Il établit une connexion TCP avec l'API à une adresse IP/port prédéfini.
- Un **handshake** a lieu avec un mot de passe pour authentification.
- Une fois connecté, le Pico écoute les commandes reçues depuis l'API.
- L'API envoie les commandes sous forme de messages JSON via TCP.



Interface Web (Frontend) :

Le HTML consiste en une page web avec les différents boutons directionnels de la voiture, chaque bouton est lié à un appel fonction sur js

Le CSS permet de placer les boutons en croix comme sur une vraie télécommande et l'intégration de différents styles css pour rendre l'interface plus agréable pour l'utilisateur.

Explication du fonctionnement des boutons de contrôle et de l'envoi des requêtes à l'API.

Des écouteurs d'événements sont mis en place (key up, key down, etc …) qui appellent une fonction sendCommand() qui se charge d'envoyer une requête HTTP POST à l'api

Description de la fonctionnalité de statut de la voiture (polling toutes les 3 secondes).

Toute les 3 secondes on envoie une requête get vers l'endpoint /api/status, on convertit la réponse obtenu en JSON et on vérifie si le Raspberry Pi Pico w est connecté, si c'est le cas on ajoute une classe connected qui vient ajouter une classe connected sinon on l'enlève. cela nous permet de savoir l'état de la connexion ce qui est utile pour indiquer à l'utilisateur que la pico est prêt à recevoir des commandes ou si une erreur s'est produite, nous appliquons la même logique avec l'endpoint /api/commandCounts pour afficher les différents compteurs sur la page

API (Backend) :

On a l'endpoint command qui reçoit des commandes du client web et transmet ces commandes via TCP au pico W qui se charge d'activer les moteurs correspondants à la commande reçue.

l'endpoint status qui vérifie l'état de la connexion entre le serveur et le pico W via la méthode GET.

Utilisation d'un clé WPA2-PSK utilisé pour se connecté au WI-FI pour établir une connexion sécurisé avec le pico w

La connection au serveur web est sécurisée grâce à un certificat SSL, on utilise le protocole HTTPS à la place de HTTP, ce qui permet de chiffrer les échanges

La connexion avec la base de données se fait avec une clé secrète stockée dans un .env

Base de Données (MongoDB) :

Nous avons 5 entrées dans la base de données (FORWARD, BACKWARD, LEFT, RIGHT, STOP) qui se présentent sous la forme de compteurs qui sont incrémentés à chaque fois que l'utilisateur, sur la page web, à condition que la voiture puisse recevoir la commande, sont incrémentés individuellement en fonction de la commande correspondante qui a été envoyée à la voiture.

Ces données sont ensuite récupérées sur la page web et affichées à l'utilisateur en temps réel.

III. Mise en Œuvre et Tests

Processus de Développement :

Notre projet s'est déroulé en quatre grandes étapes :

Conception matérielle

Nous avons sélectionné les composants (Pico W, module L298N, moteurs, batterie) et conçu le câblage sur une breadboard, montés sur un châssis LEGO modulaire. L'objectif était d'assurer stabilité, simplicité de connexion et évolutivité.

Conception logicielle

Le code a été développé en MicroPython. Il comprend la gestion du Wi-Fi, un serveur HTTP, et des fonctions de commande moteur. Une interface Web (HTML/CSS/JS) hébergée localement permet de piloter la voiture depuis un smartphone.

Intégration

Le matériel et le logiciel ont été réunis pour tester la réactivité de la voiture aux commandes Web. Plusieurs ajustements ont été faits pour améliorer la stabilité des connexions et la précision du contrôle.

Tests

Des tests unitaires et fonctionnels ont été réalisés : contrôle moteur, portée Wi-Fi, autonomie, comportement sur différents sols. Chaque test a permis d'améliorer le système.

Méthodologie de travail du groupe

Le travail de groupe a été structuré de la façon suivante :

- **GitHub Projects** pour l'organisation des tâches et le suivi de l'avancement. https://github.com/users/sean-vergauwen/projects/3
- **Repo Github** pour centraliser le code https://github.com/sean-vergauwen/elec-digi-webapp
- **Discord** pour la communication quotidienne et le partage rapide d'informations.
- **Réunions hebdomadaires** pour faire le point, répartir les tâches et ajuster les priorités.
- **Google Docs** pour la rédaction collaborative du rapport, permettant à chacun de contribuer en temps réel.

Cette organisation nous a permis d'assurer un bon rythme de développement, une collaboration efficace et une résolution rapide des problèmes.

- ## Tests et Validation :

Tests unitaires :

Nous avons réalisé un script personnalisé qui nous permet de tester tous les composants de notre PCB ainsi que le HC-SR04 et le L298N, nous avons aussi utilisé Postman pour tester l'API et ses différents endpoint. Quant à la connexion TCP de la voiture elle fût testé directement de manière "bare metal" sur celle-ci.

Simulation :

Pour la simulation nous avons utilisé Wokwi, voici le lien vers la simu de notre groupe : https://wokwi.com/projects/431390513281876993, cette simulation utilise le script de test qui peut être aussi utilisé directement sur la voiture.

Résultats des Tests :

La simulation s'est avérée concluante car elle nous a montré que notre code et nos branchements étaient corrects. Malheureusement ceux-ci nous ont permis de nous rendre compte que certaines connexions sur notre PCB étaient défectueuses car certains PINS du pico ne fonctionnent pas ce qui rend les afficheurs 7 segments et les leds inutilisables.

IV. Gestion de Projet et Collaboration

Répartition des Tâches :

La répartition des tâches au sein du groupe s'est faite de manière collaborative et évolutive. Plutôt que d'attribuer de façon rigide des rôles fixes, nous avons choisi une organisation souple permettant à chacun de contribuer à plusieurs aspects du projet, en fonction des besoins et des disponibilités.

Les grandes phases du projet, telles que la conception mécanique, la réalisation du schéma électronique et du PCB, la programmation de la carte embarquée, le développement de l'interface web, la configuration de l'API et de la base de données, ainsi que la rédaction du rapport, ont été abordées collectivement. Certaines tâches ont naturellement été prises en main par ceux qui étaient les plus à l'aise dans le domaine concerné, mais l'ensemble du groupe a participé à la validation, aux tests, aux ajustements et aux choix techniques.

Cette approche a permis à tous les membres de se familiariser avec l'ensemble du projet, d'acquérir de nouvelles compétences, et d'assurer une continuité dans le travail, même en cas d'imprévus.

Pour faciliter la coordination, nous avons utilisé plusieurs outils de gestion collaborative : un tableau Kanban pour visualiser les étapes du projet et suivre la progression des tâches sur GitHub Projects ainsi qu'un repo Git sur Github pour gérer les différentes versions du code et finalement Discord, qui a servi de canal principal de communication au quotidien.

Ce mode de fonctionnement a favorisé une répartition équilibrée de la charge de travail, tout en assurant une bonne flexibilité. Il a également renforcé l'esprit d'équipe, chaque membre ayant pu s'impliquer dans toutes les dimensions du projet.

Communication et Coordination :

Utilisation d'un groupe Discord pour pouvoir se tenir au courant sur les avancements du projet, les éventuels problèmes rencontrés et pour se fixer un moment progresser avancer sur le projet

La communication au sein du groupe a été efficace tout au long du projet grâce à l'utilisation de Discord et aussi la communication sur place lors des labos était un plus, cela a permis à chacun de savoir où on est dans l'avancement du projet

Gestion des Échéances :

Nous avons atteint avec succès les principales échéances du projet. Tout d'abord, le rapport de mi-parcours a été réalisé en parallèle avec le début de la conception du projet, et il a été rendu dans les délais. Concernant la réalisation du schéma PCB, nous l'avons également remis à temps. Nous n'avons rencontré aucun retard concernant les principales échéances et étions prêts à l'avance pour chacune d'entre elles.

- ## Bilan de la Collaboration :

Points forts :
Le point fort de notre collaboration a été surtout la communication ainsi que la répartition claire des tâches

Point faible :
Début de projet un peu désorganisé, ce qui nous a fait perdre un peu temps

Ce projet nous a permis de mettre en pratique les connaissances acquises lors des différents laboratoires d'électronique. Il nous a également montré qu'un projet concret se déroule rarement sans difficultés : nous avons dû faire face à plusieurs imprévus et rechercher activement des solutions. Pour y parvenir, nous avons appris à consulter différentes sources et à échanger nos idées afin de résoudre les problèmes ensemble. Ce travail nous a aussi permis de renforcer des soft skills important comme la communication ainsi que le travail d'équipe

V. Conclusion et Perspectives

- ## Bilan Général du Projet :

Nous avons pû réaliser presque tout ce que l'on devait, nous sommes donc assez fiers de nous. Certains points en revanche auraient pû mieux se dérouler comme par exemple l'énorme perte de temps et de budget qu'on a subi pour faire fonctionner la marche arrière.

- ## Limites du Projet :

Actuellement, le projet fonctionne et la voiture peut se déplacer grâce à l'interface web. Cependant l'une des limitations que l'on a identifié du projet est le dysfonctionnement du PCB qui ne laisse passer aucun courant dans les résistances ce qui a pour conséquence de n'allumer aucune des led ni aucun des afficheurs 7 segments. Nous avons bien vérifié les soudures, les composants utilisés, la simulation, le schéma du PCB et pourtant rien n'y fait.

- ## Perspectives d'Amélioration et Développements Futurs :

Intégration d'une caméra embarquée permettant de retransmettre en direct une vue à la première personne via l'interface web. Cette fonctionnalité offrirait une expérience de pilotage plus immersive et serait particulièrement utile lorsque la voiture est n'est plus dans notre champs de vision

Ajout de capteurs afin de mesurer et enregistrer la vitesse du véhicule en temps réel. Ces données pourraient ensuite être affichées sur l'interface web.

Enregistrement et suivi des trajets, les parcours réalisés et les distances parcourues pourraient être enregistrés localement ou envoyés vers une plateforme cloud. Cela permettrait un suivi précis des déplacements de la voiture

Ajout d'un bouton toggle pour activer ou désactiver le freinage automatique si besoin

Ajout d'une boîte de vitesses automatique, d'un différentiel et de suspensions sur la voiture.

Les diverses compétences qu'on a développé et mis en pratique au cours de ce projet peuvent être appliquées dans de nombreux projets futurs. Elles sont particulièrement utiles dans le développement embarqué, ou la maîtrise des microcontrôleurs, des capteurs et des interfaces de communication est essentielle. De plus, les connaissances acquises en gestion et manipulation de bases de données pourront être dans des projets en gestion de données qui implique la collecte, le stockage ainsi que l'analyse des données.

VI. BIBLIOGRAPHIE

- Documentation officielle Raspberry Pi PICO : raspberrypi.com
- Tutoriels MicroPython : micropython.org
- Fiches techniques des principaux éléments electroniques : datasheets.com
- Fiches techniques des moteurs LEGO : https://www.philohome.com/pf/pf.htm
- Le cours d'électronique digitale (EPHEC)
- Tutoriel pour le L298N : https://randomnerdtutorials.com/micropython-esp32-esp8266-dc-motor-l298n/
- Tutoriel pour le HC-SR04 : https://randomnerdtutorials.com/micropython-hc-sr04-ultrasonic-esp32-esp8266/
- Blogpost de création d'une voiture LEGO : https://pdwhomeautomation.blogspot.com/2012/11/raspberry-pi-powered-lego-car.html
- Notre repo Github : https://github.com/sean-vergauwen/elec-digi-webapp
- Notre Github projects : https://github.com/users/sean-vergauwen/projects/3
- Le lien vers l'interface web : https://voiture.seanvergauwen.com